#3 ed 11-18-02

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	**Araya Yasuteru, et al.**	**Examiner:**	**Unassigned**
Serial No:	**Unassigned**	**Art Unit:**	**Unassigned**
Filed:	**Herewith**	**Docket:**	**14669**
For:	**BUS PERFORMANCE EVALUATION METHOD FOR ALGORITHM DESCRIPTION**	**Dated:**	**June 1, 2001**

Assistant Commissioner for Patents
Washington, D.C. 20231

CLAIM OF PRIORITY

Sir:

Applicant in the above-identified application hereby claims the right of priority in connection with Title 35 U.S.C. § 119 and in support thereof, herewith submits a certified copy of Japanese Patent Application No. 2000-166630, filed on June 2, 2000.

Respectfully submitted,

Paul J. Esatto, Jr.
Registration No.: 30,749

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

Express Mailing Label No.: EL 915257271 US

Date of Deposit: June 1, 2001

I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. § 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents and Trademarks, Washington, D.C. 20231 on June 1, 2001.

Dated: June 1, 2001

Janet Grossman
Janet Grossman

OSP-10591 US

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２０００年　６月　２日

出願番号
Application Number: 特願２０００－１６６６３０

出願人
Applicant (s): 日本電気アイシーマイコンシステム株式会社

２００１年　３月　２日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号　出証特２００１－３０１３３８０

【書類名】　特許願
【整理番号】　01211132
【提出日】　平成12年 6月 2日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 17/50
【発明の名称】　アルゴリズム記述におけるバスの性能評価方法
【請求項の数】　8
【発明者】
【住所又は居所】　神奈川県川崎市中原区小杉町一丁目４０３番５３　日本電気アイシーマイコンシステム株式会社内
【氏名】　荒谷　泰輝
【発明者】
【住所又は居所】　神奈川県川崎市中原区小杉町一丁目４０３番５３　日本電気アイシーマイコンシステム株式会社内
【氏名】　丸山　勇一
【特許出願人】
【識別番号】　000232036
【氏名又は名称】　日本電気アイシーマイコンシステム株式会社
【代理人】
【識別番号】　100108578
【弁理士】
【氏名又は名称】　高橋　詔男
【代理人】
【識別番号】　100064908
【弁理士】
【氏名又は名称】　志賀　正武
【選任した代理人】
【識別番号】　100101465
【弁理士】

【氏名又は名称】　青山　正和

【選任した代理人】

【識別番号】　100108453

【弁理士】

【氏名又は名称】　村山　靖彦

【手数料の表示】

【予納台帳番号】　008707

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9901153

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　アルゴリズム記述におけるバスの性能評価方法

【特許請求の範囲】

【請求項１】　アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、

前記ソースを、前記バスにデータ転送が生じた際に、特定の評価関数を実行するように修正することにより、アーキテクチャー設計を行うためのシミュレーションプラットホームを構築することを特徴とするアルゴリズム記述における性能評価方法。

【請求項２】　前記シミュレーションプラットホームを用いて、シミュレーションを行うことにより、処理レートに対するバストラフィックを算出し、前記バスの性能評価を上位設計の段階において行うことを特徴とする請求項１に記載のアルゴリズム記述における性能評価方法。

【請求項３】　前記評価対象となるバスの性能評価結果を前記アルゴリズム検証で用いたソースにフィードバックすることにより、上位設計の段階において、アーキテクチャー設計を行うことを特徴とする請求項１または２に記載のアルゴリズム記述における性能評価方法。

【請求項４】　アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、

前記ソースを用いてアーキテクチャー設計を行うためのシミュレーションプラットフォームを構築し、

前記構築したシミュレーションプラットフォームを用いて評価対象となるバスの性能評価を行い、その評価結果に基づき前記ソースを修正して前記シミュレーションプラットフォームを再構築し、

前記再構築されたシミュレーションプラットフォームを用いて前記性能評価を行うことを特徴とするアルゴリズム記述におけるバスの性能評価方法。

【請求項５】　前記ソースを用いてアーキテクチャー設計を行うためのシミ

ュレーションプラットフォームを構築するために、

前記ソースを所定の単位で解析することによりハードウェアとソフトウェア記述に分け、

前記バスのトラフィックをカウントする評価関数を作成し、

前記バスを使用したデータ転送が発生する都度、前記作成された評価関数を実行するように前記ソース構文を修正することを特徴とする請求項４に記載のアルゴリズム記述におけるバスの性能評価方法。

【請求項６】　前記性能評価を行うために、

前記評価関数に基づいて性能評価を行い、

最終的に算出された処理レートに対するバストラフィックにより、前記ハードウェアとソフトウェアの切り分けの最適化を行うことを特徴とする請求項４に記載のアルゴリズム記述におけるバスの性能評価方法。

【請求項７】　アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、

前記バスのトラフィックをカウントする評価関数を作成する工程と、

前記ソースを構文解析しながら行単位で順次読み込む工程と、

前記ソースの記述が評価対象となるバスに乗るあらかじめ定義された変数へのデータの書き込みを表しているか否かを判定する工程と、

前記判定の結果、データの書き込みを表していた場合、前記作成した評価関数をその直前もしくは直後に埋め込み前記ソースの記述を修正する工程と、

前記読み込んだソース構文が最後の行になるまで前記各工程の処理を繰り返し、コンパイルしてアーキテクチャー設計を行うためのシミュレーションプラットフォームを構築し、これを実行して評価対象となるバスのトラフックを算出する工程と、

既知の処理レートに対するバストラフィックを算出して評価対象となるバスの性能評価を行う工程とを備えたことを特徴とするアルゴリズム記述におけるバスの性能評価方法。

【請求項８】　前記評価対象となるバスに乗るそれぞれの変数のビット幅を

ｎ、前記評価対象のバスを構成するビット幅をｍ（但し、ｎ≧ｍ）とした場合、前記処理レートに対するバストラフィックは、データ転送回数にｎ／ｍ倍したものを処理レートで除算することによって算出して性能評価を行うことを特徴とする請求項７に記載のアルゴリズム記述におけるバスの性能評価方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、アーキテクチャー設計を上位設計の段階で行うために、アルゴリズム検証用に、汎用高級言語（ＣやＣ$^{++}$）で記述したソースを用いて、ハードウェア(以下、単にＨ／Ｗという)、ソフトウェア(以下、単にＳ／Ｗという）をつなぐバスの性能評価を行う、アルゴリズム記述におけるバスの性能評価方法に関する。

【０００２】

【従来の技術】

半導体技術の進歩に伴い、ボードに多数のＬＳＩチップを配置してシステムを具現化する形態から、単体のＬＳＩチップでシステムを具現化する形態が増えてきた。そのため、従前のＬＳＩでは、外部端子として接続していた信号が内部信号となり、ＬＳＩに内蔵されている。従って、従来のようにボードを使用したシステム検証を行うためには、内部信号をすべて外部端子とした評価用のＬＳＩを別途製造しなければならず、実質以前のようなボードを使用した検証を行うことが困難になってきた。

【０００３】

図１１に従来のＬＳＩの設計開発の流れをフローチャートで示す。ロジック回路、システム及びデバイスを製造する際、通常は、まず、ハードウェア、ソフトウェアを意識しないシミュレーションプラットホームを構築し(ステップＢ１)、アルゴリズムが正しいか否かを検証する(ステップＢ２)。次に、構築したシミュレーションプラットホームから、ハードウェア、ソフトウェアに切り分ける。このハードウェア、ソフトウェアへの切り分けは、経験的に行われてきた。

【０００４】

Ｈ／Ｗ設計において、まずアルゴリズムと等価な機能を持つソースを一般的にはＨＤＬ（Hardware Description Language）言語で記述し、回路合成を行う（ステップＢ３）。次に、作成したソースが正常に動作するか否かを検証する（ステップＢ４）。Ｓ／Ｗ設計において、まずアルゴリズムと等価な機能を持つソースを一般的には、各ＣＰＵに依存したプログラミング言語で記述する（ステップＢ５）。次に、作成したソースが正常に動作するか否かを検証する（ステップＢ６）。最後に、ハードウェア、ソフトウェアを結合した協調検証が行われる（ステップＢ７）。

【０００５】

【発明が解決しようとする課題】

上述したように、ＬＳＩを製造するにあたり、実際に作り込む前にシステムシミュレーションを行い、バスの性能評価を行い、ハードウェア、ソフトウェアの切り分けを行うアーキテクチャー設計をシミュレーションにより行うことが必須条件となってきた。従来、システムシミュレーションは上述したようにハードウェア、ソフトウェアを統合した協調検証により行われてきた。

【０００６】

近年、製造する回路規模の増加と共に、作成する回路のコード数が多大なものとなってきた。従って、ＨＤＬ言語、各ＣＰＵに依存するプログラミング語で記述された協調検証のシミュレーションスピードが非常に遅くなり、実質、協調検証によるアーキテクチャー設計を行うことが困難になってきた。

また、協調検証によりアーキテクチャー設計を行った場合、ハードウェア、ソフトウェアをつなぐバスの性能評価結果により、ハードウェア、ソフトウェアをつなぐバスの変更等が生じた場合、ＨＤＬ言語、各ＣＰＵに依存したプログラミング言語を修正する必要がある（図１１、ステップＢ８、Ｂ９）。

【０００７】

この場合、作成する回路のコード数の増加に従い、作成する回路記述がより複雑なものになってきたため回路記述の修正が複雑なものとなり、協調検証からのフィードバックに要する時間と費用が多大なものとなっている。従って、回路規模が増大し、開発期間が短縮されてきている近年において、ハードウェア、ソフ

トウェアをつなぐバスの性能評価を上位設計で行い、アーキテクチャー設計を上位設計の段階で行うことが設計開発において必須条件となってきた。

【０００８】

本発明は上記事情に鑑みてなされたものであり、アルゴリズム設計で作成した汎用高級言語で記述されたソースをハードウェア、ソフトウェアに切り分けた後、バスのトラフィックをカウントする評価関数を作成して、バスにデータ(変数)が乗る毎に、その評価関数を実行するように、ソースを修正し、その後、性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減を可能としたアルゴリズム記述におけるバスの性能評価方法を提供することを目的とする。

【０００９】

【課題を解決するための手段】

上記した課題を解決するために請求項１に記載の発明は、アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、ソースを、バスにデータ転送が生じた際に、特定の評価関数を実行するように修正することにより、アーキテクチャー設計を行うためのシミュレーションプラットホームを構築することとした。

このことにより、アルゴリズム設計で用いた、例えば、Ｃ言語、Ｃ++言語等汎用高級言語で記述されたソースを、評価対象となるバスに乗る変数にデータの書き込みが生じた際に、一定の値をインクリメントする特定の評価関数を実行するように修正し、シミュレーションプラットホームを構築するため、設計開発フローの中で、上位設計の段階において、ハードウェア、ソフトウェアをつなぐバスの処理レートに対するバストラフィックを算出し、バスの性能評価を行うことができる。

【００１０】

請求項２に記載の発明は、請求項１に記載のアルゴリズム記述における性能評価方法において、シミュレーションプラットホームを用いて、シミュレーションを行うことにより、処理レートに対するバストラフィックを算出し、バスの性能評価を上位設計の段階において行うこととした。

このことにより、構築されたシミュレーションプラットホームを用いて性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減が可能となる。

【００１１】

請求項３に記載の発明は、請求項１または２に記載のアルゴリズム記述における性能評価方法において、評価対象となるバスの性能評価結果を前記アルゴリズム検証で用いたソースにフィードバックすることにより、上位設計の段階において、アーキテクチャー設計を行うこととした。

このことにより、上位設計の段階で、評価対象となるバスの性能評価結果をフィードバックするため、設計開発フローの上位設計段階において、アーキテクチャー設計を行うことができる。

【００１２】

請求項４に記載の発明は、アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、ソースを用いてアーキテクチャー設計を行うためのシミュレーションプラットフォームを構築し、構築したシミュレーションプラットフォームを用いて評価対象となるバスの性能評価を行い、その評価結果に基づきソースを修正してシミュレーションプラットフォームを再構築し、再構築されたシミュレーションプラットフォームを用いて性能評価を行うこととした。

このことにより、アルゴリズム設計で用いた、例えば、Ｃ言語、Ｃ++言語等汎用高級言語で記述されたソースを、評価対象となるバスに乗る変数にデータの書

き込みが生じた際に、一定の値をインクリメントする特定の評価関数を実行するように修正し、シミュレーションプラットホームを構築するため、設計開発フローの中で、上位設計の段階において、ハードウェア、ソフトウェアをつなぐバスの処理レートに対するバストラフィックを算出し、バスの性能評価を行うことができる。

【００１３】

請求項５に記載の発明は、請求項４に記載のアルゴリズム記述におけるバスの性能評価方法において、ソースを用いてアーキテクチャー設計を行うためのシミュレーションプラットフォームを構築するために、ソースを所定の単位で解析することによりハードウェアとソフトウェア記述に分け、バスのトラフィックをカウントする評価関数を作成し、バスを使用したデータ転送が発生する都度、前記作成された評価関数を実行するように前記ソース構文を修正することとした。

このことにより、上位設計の段階で、評価対象となるバスの性能評価結果をフィードバックするため、設計開発フローの上位設計段階において、アーキテクチャー設計を行うことができる。また、アルゴリズム設計で用いたＣ言語、Ｃ++言語等汎用高級言語で記述されたソースによりアーキテクチャー設計を行うため、アーキテクチャー設計に要するシミュレーション時間を大幅に短縮することができる。

【００１４】

請求項６に記載の発明は、請求項４に記載に記載のアルゴリズム記述におけるバスの性能評価方法において、性能評価を行うために、評価関数に基づいて性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、ハードウェアとソフトウェアの切り分けの最適化を行うこととした。

このことにより、例えば、Ｃ言語、Ｃ++言語等汎用高級言語で記述されたソースによりアーキテクチャー設計を行うため、アーキテクチャー設計によるフィードバックを容易化でき、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証における切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減を可能とする

【００１５】

請求項７に記載の発明は、アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、バスのトラフィックをカウントする評価関数を作成する工程と、ソースを構文解析しながら行単位で順次読み込む工程と、ソースの記述が評価対象となるバスに乗るあらかじめ定義された変数へのデータの書き込みを表しているか否かを判定する工程と、判定の結果、データの書き込みを表していた場合、作成した評価関数をその直前もしくは直後に埋め込み前記ソースの記述を修正する工程と、読み込んだソース構文が最後の行になるまで前記各工程の処理を繰り返し、コンパイルしてアーキテクチャー設計を行うためのシミュレーションプラットフォームを構築し、これを実行して評価対象となるバスのトラフックを算出する工程と、既知の処理レートに対するバストラフィックを算出して評価対象となるバスの性能評価を行う工程とを備えることとした。

このことにより、アルゴリズム設計で作成した汎用高級言語で記述されたソースをハードウェア、ソフトウェアに切り分けた後、バスのトラフィックをカウントする評価関数を作成して、バスにデータ(変数)が乗る毎に、その評価関数を実行するように、ソースを修正し、その後、性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排するることができ、設計ターンアラウンドタイムの大幅な削減が可能となる。

【００１６】

請求項８に記載の発明は、請求項７に記載のアルゴリズム記述におけるバスの性能評価方法において、評価対象となるバスに乗るそれぞれの変数のビット幅をｎ、評価対象のバスを構成するビット幅をｍ（但し、ｎ≧ｍ）とした場合、処理レートに対するバストラフィックは、データ転送回数にｎ／ｍ倍したものを処理レートで除算することによって算出して性能評価を行うこととした。

このことにより、評価対象となるバスに乗る変数のビット幅が異なり、且つ評価対象となるバスのビット幅より大きい場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００１７】

【発明の実施の形態】

図１は、本発明に従うＬＳＩの設計開発の流れをフローチャートで示した図である。図 1において、アーキテクチャー設計を上位設計の段階において行うために、従来から使用されていた設計開発フロー（図１１）に、シミュレーションプラットホームを構築する工程と、性能評価工程（ステップＡ１５）が追加されている。以下にその詳細を記述する。

【００１８】

まずアルゴリズム設計を行い(ステップＡ１)、製造するロジック回路、デバイス及びシステムのアルゴリズムが正しいか否かをアルゴリズム検証により検証する(ステップＡ２)。

次に、アルゴリズム設計において使用されたソースを用いて、アーキテクチャー設計を行うためのシミュレーションプラットホームを構築する。ここでは、まず、ハードウェア、ソフトウェアを切り分ける(ステップＡ３)。次に、評価関数を作成する(ステップＡ４)。ここで、評価関数は、一定量をカウントする関数であれば良い。

【００１９】

次に、ハードウェア、ソフトウェアをつなぐバスに乗る変数を選択する(ステップＡ５)。そして、アルゴリズム設計で用いたソースを、評価対象となるバスに乗る変数にデータの書き込みが生じた際、すなわち評価対象となるバスにデータ転送が生じた際に、作成した評価関数を実行するように修正し、アーキテクチャー設計を行うためのシミュレーションプラットホームを構築する(ステップＡ６)。

【００２０】

次に、構築したシミュレーションプラットホームを用いて、評価対象となるバスの性能評価を行う。まず、構築したシミュレーションプラットホームを用いて、検証を行う(ステップＡ７)。そして、作成した評価関数により方法は異なるが、バストラフィックを算出する(ステップＡ８)。ここでは、メイン関数に要求される処理レートが既知であることから、評価対象となるバスの処理レートに対するバストラフィックを算出する(ステップＡ９)。

算出した処理レートに対するバストラフィックにより、ハードウェア、ソフトウェアの切り分け及びバス構成の妥当性の検討ができる。検討後、バスの変更等が必要な場合、汎用高級言語、例えば、Ｃ言語やC++言語で記述されたソースを修正して、再度、シミュレーションプラットホームの構築、性能評価を行う。このように評価対象となるバスの性能評価結果をフィードバックすることにより(ステップＡ１６)、上位設計の段階におけるアーキテクチャー設計を実現する。

【００２１】

アーキテクチャー設計を行った後、ハードウェア、ソフトウェアそれぞれの設計フローに入る。Ｈ／Ｗ設計において、まずアルゴリズムと等価な機能を持つソースを一般的にはＨＤＬ言語で記述し、回路合成を行う(ステップＡ１０)。次に、作成したソースが正常に動作するか否かを検証する(ステップＡ１１)。Ｓ／Ｗ設計において、まずアルゴリズムと等価な機能を持つソースを一般的には、各ＣＰＵに依存したプログラミング言語で記述する(ステップＡ１２)。そして、作成したソースが正常に動作するか否かを検証する(ステップＡ１３)。

【００２２】

最後に、ハードウェア、ソフトウェアを結合した協調検証を行う(ステップＡ１４)。アーキテクチャー設計は、本発明の設計開発フローにおいては、既に上位設計の段階において終了しているため、協調検証は、動作確認を行うのみのシミュレーションと位置づけ、設計を行わない。従って、協調検証からのフィードバックを行う必要が無くなる。

【００２３】

図２に本発明の一実施形態をリスト形式で示す。図２を参照すれば、評価対象となるバスに乗る変数として、ａ、ｂ、ｃの3つの変数を持ち、いずれもグロー

バル定義されている。また、評価関数ＢＵＳ０（）と、ローカル変数ｂｕｓ０を持つ。ここでは、評価関数は、スタティック変数ｉをインクリメントし、戻り値とする関数である。ローカル変数ｂｕｓ０は、評価関数ＢＵＳ０（）からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。また、評価対象となるバスに乗る変数ａ、ｂ、ｃのビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。

【００２４】

図２に示す実施形態では、評価対象となるバスに乗る変数ａ、ｂ、ｃにデータの書き込みが行われるとき、必ず評価関数ＢＵＳ０（）を実行するように、直後に評価関数ＢＵＳ０（）が埋め込まれている。評価関数ＢＵＳ０（）は、実行される度に、スタティック変数ｉを1だけインクリメントし、戻り値とするため、評価関数ＢＵＳ０（）からの戻り値は、評価対象となるバスに乗る変数のデータ書き込み回数を表すことになる。

【００２５】

評価対象となるバスに乗る変数のデータ書き込み回数は、評価対象となるバスのデータ転送、すなわちバストラフィックを表している。メイン関数に要求される処理レートが既知であるため、下記式１により処理レートに対するバストラフィックを算出し、性能評価を行うことができる。

（処理レートに対するバストラフィック）

＝（データ転送回数）／（処理レート）…　　（１）

算出した処理レートに対するバストラフィックにより、ハードウェア、ソフトウェアをつなぐバスを変更する場合、アルゴリズム設計で用いられたソースを修正し、再度シミュレーションプラットホームを構築する。

【００２６】

図３に再度構築したシミュレーションプラットホームの一例を示す。図３を参照すれば、再度構築したシミュレーションプラットホームの一例は、図２に示す実施形態における評価対象となるバスの性能評価結果により、変数ｂを評価対象となるバスに乗らない変数としている。従って、評価対象となるバスに乗る変数として、ａ、ｃの２つの変数のみを持つ。変数ｂはバスに乗らないので、変数ｂ

にデータの書き込みが行われる直後には、評価関数ＢＵＳ０（）が埋め込まれておらず、変数ａ、ｃにデータの書き込みが行われるときには、直後に評価関数ＢＵＳ０（）が埋め込まれ、評価関数ＢＵＳ０（）を必ず実行するようになっている。

シミュレーションプラットホームを再度構築した後、シミュレーションにより、検証を行う。再度、式１により処理レートに対するバストラフィックを算出し、評価対象となるバスの性能評価を行う。

【００２７】

以上説明のように、評価対象となるバスのデータレートに対する平均バストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００２８】

図４に、図１に示す実施形態の動作がフローチャートで示されている。

図４を参照すると、まず特定の評価関数を作成する(ステップＣ１)。評価関数は、ある一定の値をインクリメントする関数であれば何でも良い。次に、アルゴリズム設計で用いたＣ言語、もしくはC++言語で記述されたソースを、構文解析を行いながら1行ずつ読み込む(ステップＣ２)。

【００２９】

次に、読み込んだソースの記述が、評価対象となるバスに乗る変数へのデータの書き込みを表しているか否かを判定する(ステップＣ３)。評価対象となるバスに乗る変数へのデータの書き込みを表している場合、その直後に作成した評価関数を埋め込む(ステップＣ４)。この作業により、評価対象となるバスに乗る変数にデータの書き込みが行われた場合、すなわち評価対象となるバスにデータ転送が生じた場合に、必ず作成した評価関数を実行するようになる。

【００３０】

以上の作業を読み込んだソースの記述が、アルゴリズム設計で用いたソースの最後の行まで行い、アルゴリズム設計で用いたソースを修正する(ステップＣ５)。アルゴリズム設計で用いられたソースの最後の行まで修正が行われた後、コンパイルをして、アーキテクチャー設計を行うためのシミュレーションプラットホ

ームを構築する(ステップＣ６)。次に、それを実行し、評価対象となるバスのバストラフィックを算出する(ステップＣ７)。そして、メイン関数に要求される処理レートが既知であることから、処理レートに対するバストラフィックを算出し、評価対象となるバスの性能評価を行う(ステップＣ８)。

【００３１】

図５乃至図１０は本発明の他の実施形態をリスト形式で示した図である。以下、本発明の他の実施形態につき、図２に示す実施形態と対比しながら詳細に説明する。

【００３２】

図５に示す実施形態は、評価対象となるバスに乗る変数として、a、b、cの3つの変数を持ち、いずれもローカル定義されている。また、評価関数ＢＵＳ０()と、ローカル変数ｂｕｓ０()を持つ。ここでは、評価関数は、スタティック変数ｉをインクリメントし、戻り値とする関数である。ローカル変数ｂｕｓ０()は、評価関数ＢＵＳ０()からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【００３３】

ここでは、評価対象となるバスに乗る変数a、b、cのビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとし、評価対象となるバスに乗る変数が、グローバル定義ではなく、ローカル定義されていることが図２に示す実施形態と異なる。ここで性能評価を行う場合、上記式１により処理レートに対するバストラフィックを算出して行う。

【００３４】

以上説明のように、評価対象となるバスに乗る変数がグローバル定義ではなく、ローカル定義されている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００３５】

図６に示す実施形態は、評価対象となるバスに乗る変数として、ａ、ｂ、ｃの３つの変数を持ち、いずれもグローバル定義されている。また、評価関数ＢＵＳ

０（）と、ローカル変数ｂｕｓ０（）を持つ。ここでは、評価関数は、スタティック変数ｉをインクリメントし、戻り値とする関数である。ローカル変数ｂｕｓ０（）は、評価関数ＢＵＳ０（）からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【００３６】

評価対象となるバスに乗る変数ａ、ｂ、ｃのビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。ここで、評価対象となるバスに乗る変数にデータの書き込みが生じた際、すなわち評価対象となるバスにデータ転送が生じた際に、必ず評価関数を実行するように直前に評価関数ＢＵＳ０（）が埋め込まれている点が図２に示す実施形態とは異なる。ここで性能評価を行う場合、式１により処理レートに対するバストラフィックを算出して行う。

【００３７】

以上説明のように、評価対象となるバスに乗る変数にデータの書き込みが生じた際、すなわち評価対象となるバスにデータ転送が生じた際に、必ず評価関数を実行するように直前に評価関数ＢＵＳ０（）が埋め込まれている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００３８】

なお、ここで、評価対象となるバスに乗る変数ａ、ｂ、ｃのビット幅がすべて同一で、評価対象となるバスのビット幅より大きい場合、例えば、評価対象となるバスに乗る変数a、b、cのビット幅がｎビット、評価対象となるバスのビット幅がｍビット（但し、ｎ≧ｍ）の場合について説明する。具体的に、ｎ＝３２、ｍ＝８を想定する。

この例において、評価対象となるバスに乗る変数にデータの書き込みが行われた場合、評価対象となるバスにデータ転送は、ｎ／ｍ回、すなわち４回行われる。従って、性能評価を行う場合、下記式２により処理レートに対するバストラフィックを算出して行う。

（処理レートに対するバストラフィック）

＝（実行回数）×４／（処理レート）　…　（２）

【００３９】

次に、図７以降を参照して本発明の更に他の実施形態について説明する。

図７は、評価対象となるバスに乗る変数として、ａ、ｂの２つの変数を持ち、いずれもグローバル定義されている。また、評価関数ＢＵＳ１（）と、ローカル変数ｂｕｓ１を持つ。ここで、評価関数は、引数としてｂｉｔとｂｕｓを持ち、スタティック変数ｉを(ｂｉｔ／ｂｕｓ)分インクリメントし、戻り値とする関数である。ローカル変数ｂｕｓ１は、評価関数ＢＵＳ１（）からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【００４０】

ここでは、評価対象となるバスに乗る変数aのビット幅は３２ビット、評価対象となるバスに乗る変数ｂのビット幅は１６ビットとする。また、評価対象となるバスのビット幅は８ビットとする。

評価対象となるバスに乗る変数のビット幅が異なり、評価対象となるバスのビット幅より大きい点で図２に示す実施形態とは異なる。また、評価関数が引数としてｂｉｔとｂｕｓを持ち、スタティック変数ｉを（ｂｉｔ／ｂｕｓ）分インクリメントし、戻り値とする関数であるという点も図２に示す実施形態と異なる。ここで性能評価を行う場合、上述した式１により処理レートに対するバストラフィックを算出して行う。

【００４１】

以上説明のように、評価対象となるバスに乗る変数のビット幅が異なり、且つ評価対象となるバスのビット幅より大きい場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００４２】

図８に示す実施形態は、評価対象となるバスに乗る変数として、ａ、ｂの２つの変数を持ち、いずれも配列で定義されている。また、評価関数ＢＵＳ２（）と、ローカル変数ｂｕｓ２を持つ。更に、評価関数は、引数としてｅｌｅを持ち、

スタティック変数ｉをｅｌｅ分インクリメントし、戻り値とする関数である。ローカル変数ｂｕｓ２は、評価関数ＢＵＳ２（）からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【００４３】

ここでは、評価対象となるバスに乗る変数ａ、ｂのビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。また、評価対象となるバスに乗る変数は、配列で定義されており、メイン関数内においてアドレス転送が行われている点が図２に示す実施形態と異なる。更に、評価関数が引数としてｅｌｅを持ち、スタティック変数ｉをｅｌｅ分インクリメントし、戻り値とする関数であるという点も図２に示す実施形態と異なる。

ここで性能評価を行う場合、上述した（式１）により処理レートに対するバストラフィックを算出して行う。

【００４４】

以上説明のように、評価対象となるバスに乗る変数が配列で定義され、メイン関数内でアドレス転送が行われている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００４５】

図９に示す実施形態は、評価対象となるバスに乗る変数として、ａ、ｂの２つの変数を持ち、いずれもグローバル定義されている。また、評価関数ＢＵＳ３（）と、グローバル変数ｂｕｓを持つ。評価関数は、グローバル変数ｂｕｓをインクリメントする関数である。グローバル変数ｂｕｓは、評価対象となるバスのデータ転送回数を表す。

ここでは、評価対象となるバスに乗る変数ａ、ｂのビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。評価関数内でインクリメントする変数がグローバル定義されている点が図２に示す実施形態と異なる。

ここで性能評価を行う場合、滋養術した式１により処理レートに対するバスト

ラフィックを算出して行う。

【００４６】

以上説明のように、評価関数内でインクリメントする変数がグローバル定義されている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００４７】

図１０に、シミュレーションプラットホームをC++言語で記述したソースにより構築した例が示されている。

図１０に示す実施形態は、ＢＵＳクラス定義された評価対象となるバスｂｕｓ０、ｂｕｓ１、ｂｕｓ２と、グローバル定義された評価対象となるバスｂｕｓ０に乗る変数ａと、ローカル定義された評価対象となるバスｂｕｓ１に乗る変数ｂと、配列で定義された評価対象となるバスｂｕｓ２に乗る変数ｃを持つ。

【００４８】

また、ここでは、ＢＵＳクラスを持つ。ＢＵＳクラス内には、メンバ変数ｉを１インクリメントする関数ｃｏｕｎｔ（）と、引数ｂｉｔ、ｂｕｓを持つ。更に、メンバ変数ｉを(ｂｉｔ／ｂｕｓ)分インクリメントする関数Ｃｏｕｎｔ＿ｂｉｔ（）と、引数ｅｌｅを持ち、メンバ変数ｉをｅｌｅ分インクリメントする関数ｃｏｕｎｔ＿ｈａｉｒｅｔｕ（）が定義されている。従って、ＢＵＳクラス内のメンバ変数ｉは、評価対象となるバスのデータ転送回数を表す。

【００４９】

評価対象となるバスに乗る変数ａ、ｃのビット幅は、すべて同一とし、評価対象となるバスそれぞれｂｕｓ０、ｂｕｓ１のビット幅より小さいものとする。また、評価対象となるバスに乗る変数ｂのビット幅は３２ビット、評価対象となるバスｂｕｓ１のビット幅は８ビットとする。

ここでは、ソースがC++言語で記述されている点で上述した図２を含む各実施形態と異なり、性能評価を行う場合、（式１））により処理レートに対するバストラフィックを算出して行う。

【００５０】

以上説明のように、シミュレーションプラットホームをC++言語で記述したソースにより構築した場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【００５１】

以上説明のように本発明は、アルゴリズム設計で使用されるＣ言語、C++言語等汎用高級言語で記述されたソースを、評価対象となるハードウェア、ソフトウェアをつなぐバスにデータ転送が生じた際に、特定の評価関数を実行するように修正することでアヘクーキテクチャ設計を行うためのシミュレーションプラットフォームを構築し、ここで構築されたシミュレーションプラットホームを用いて、シミュレーションを行うことにより、処理レートに対するバストラフィックを算出し、バスの性能評価を上位設計の段階において行うものである。

【００５２】

また、評価対象となるバスの性能評価結果をアルゴリズム検証で用いたソースにフィードバックすることにより、上位設計の段階において、アーキテクチャー設計を行うことができ、アーキテクチャー設計に要するシミュレーション時間を短縮することができる。更に、ハードウェア、ソフトウェアを結合した協調検証を動作確認のみを行うシミュレーションと位置づけ、実際の設計を行わないことにより、ロジック回路、システム及びデバイスの設計開発全体に要する時間と費用を大幅に削減することができる。

【００５３】

【発明の効果】

以上説明のように本発明によれば、アルゴリズム設計で用いた、例えば、Ｃ言語、C++言語等汎用高級言語で記述されたソースを、評価対象となるバスに乗る変数にデータの書き込みが生じた際に、一定の値をインクリメントする特定の評価関数を実行するように修正することによりシミュレーションプラットホームを構築するため、設計開発フローの中で、上位設計の段階において、ハードウェア、ソフトウェアをつなぐバスの処理レートに対するバストラフィックを算出し、バスの性能評価を行うことができる。

また、上位設計の段階で、評価対象となるバスの性能評価結果をフィードバックするため、設計開発フローの上位設計段階において、アーキテクチャー設計を行うことができる。

更に、アルゴリズム設計で用いたＣ言語、Ｃ++言語等汎用高級言語で記述されたソースによりアーキテクチャー設計を行うため、アーキテクチャー設計に要するシミュレーション時間を大幅に短縮することができる。一般的にＣ言語、Ｃ++言語は、ＨＤＬ言語に比較してシミュレーションスピードが約１０００倍速い。

【００５４】

また本発明によれば、Ｃ言語、Ｃ++言語等汎用高級言語で記述されたソースによりアーキテクチャー設計を行うため、アーキテクチャー設計によるフィードバックを容易化でき、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減を可能とする。更に、Ｃ言語、Ｃ++言語等汎用高級言語は、ＨＤＬ言語、アセンブラ言語と比較して少ないコード数での記述が可能であり、変更、修正が容易にできる。

【００５５】

また、ハードウェア、ソフトウェアを結合した協調検証を動作確認のみを行うシミュレーションと位置づけ、設計を行わないため、協調検証からのフィードバックを無くし、ＲＴＬでのシミュレーション回数を削減できるため回路設計全体に要する時間と費用を大幅に削減できる。このことは、製造する回路が大きければ大きいほど、その効果は顕著に現れる。

【図面の簡単な説明】

【図１】　本発明に従うＬＳＩの設計開発の流れをフローチャートで示した図である。

【図２】　本発明の一実施形態をリスト形式で示した図である。

【図３】　図２に示す本発明実施形態において再構築したシミュレーションプラットホームの一例を示す図である。

【図４】　図２に示す実施形態における作業手順をフローチャートで示した

図である。

【図５】　本発明の他の実施形態をリスト形式で示した図である。

【図６】　本発明の更に他の実施形態をリスト形式で示した図である。

【図７】　本発明の更に他の実施形態をリスト形式で示した図である。

【図８】　本発明の更に他の実施形態をリスト形式で示した図である。

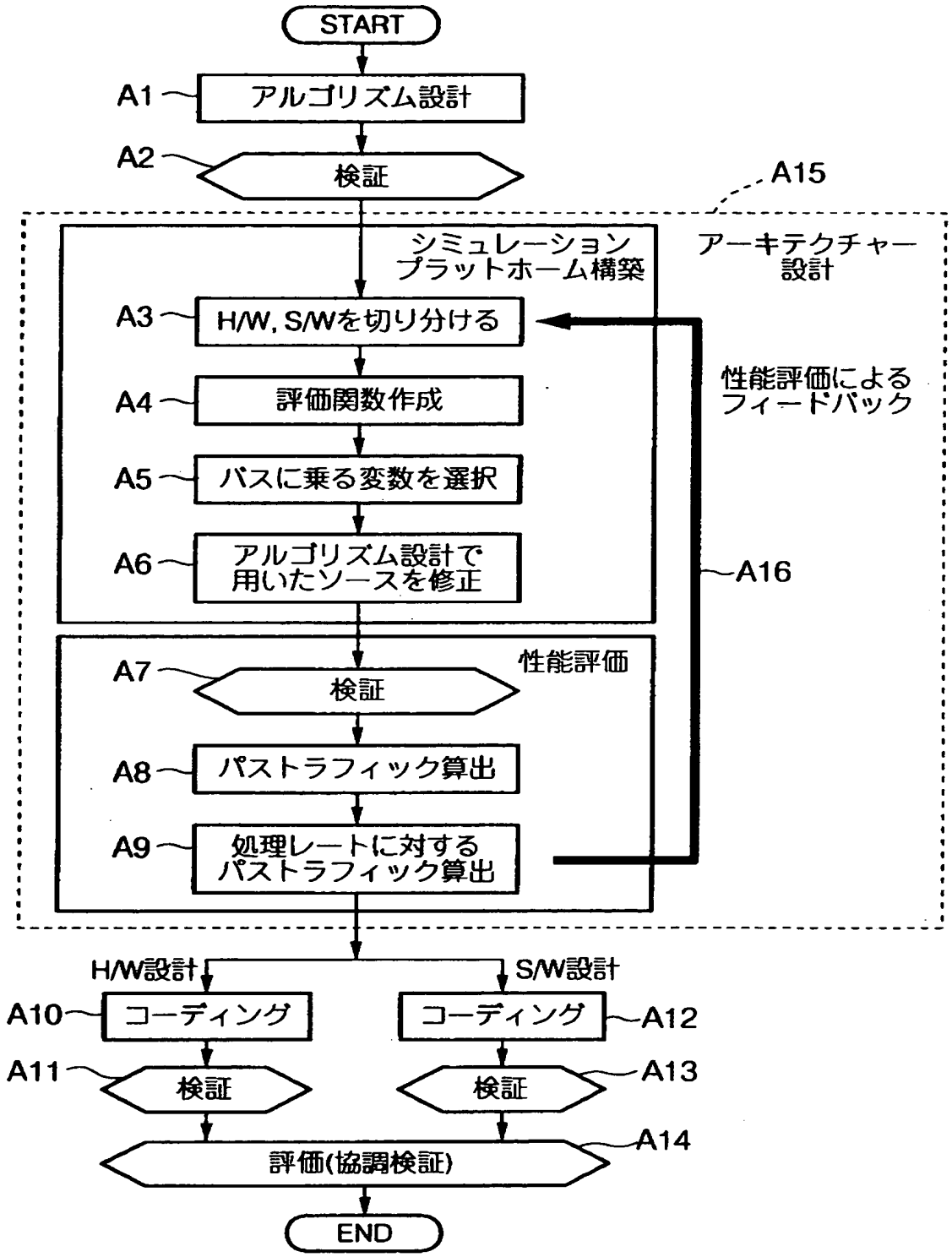
【図９】　本発明の更に他の実施形態をリスト形式で示した図である。

【図１０】　本発明の更に他の実施形態をリスト形式で示した図である。

【図１１】　従来のＬＳＩの設計開発の流れをフローチャートで示した図である。

【書類名】　図面

【図1】


START
A1
アルゴリズム設計
A2
検証
A15
シミュレーション
プラットホーム構築
アーキテクチャー
設計
A3
H/W, S/Wを切り分ける
性能評価による
フィードバック
A4
評価関数作成
A5
パスに乗る変数を選択
A6
アルゴリズム設計で
用いたソースを修正
A16
性能評価
A7
検証
A8
パストラフィック算出
A9
処理レートに対する
パストラフィック算出
H/W設計
S/W設計
A10
コーディング
コーディング
A12
A11
検証
検証
A13
A14
評価(協調検証)
END

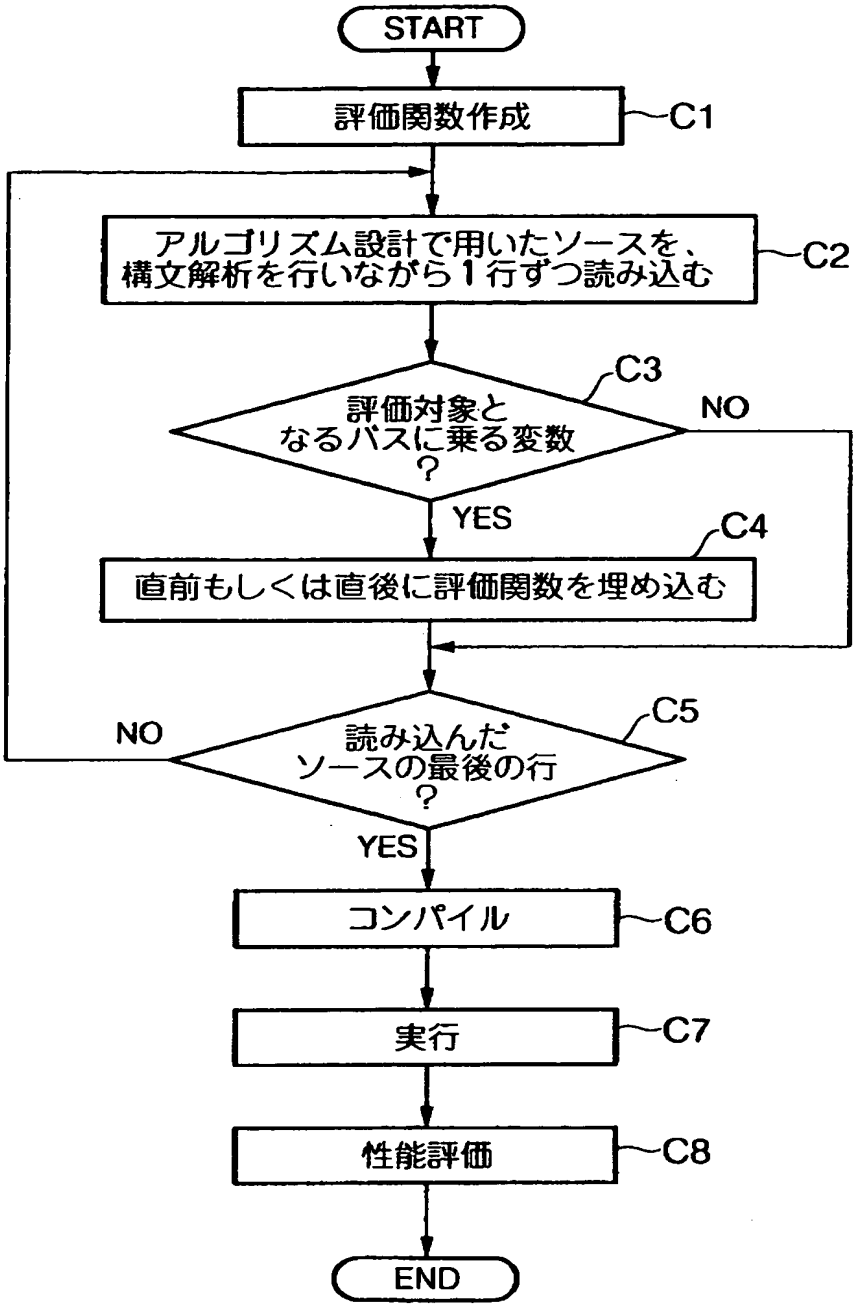
【図２】

```
#include <stdio.h>

int BUS0(void);

int a;                          /* 評価対象となるパスに乗る変数 */
int b;                          /* 評価対象となるパスに乗る変数 */
int c;                          /* 評価対象となるパスに乗る変数 */

/* メイン関数 */
void main(void)
{
 int bus0;
 .
 .
 .
 a=0;                           /* パスに乗る変数に書き込みが発生した場合 */
 bus0=BUS0();                   /* 直後に評価関数を埋め込む */
 .
 .
 .
 b=1;
 bus0=BUS0();
 .
 .
 .
 c=10;
 bus0=BUS0();
 .
 .
 .
 a=3;
 bus0=BUS();
 b=4;
 bus0=BUS();

 printf("BUS Traffic=%d[回]¥n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
 static int i=0;
 i++;                           /*スタティック変数iをインクリメント */
 return i;
}
```

【図３】

```
#include <stdio.h>

int BUS0(void);

int a;                          /* 評価対象となるバスに乗る変数 */
int b;
int c;                          /* 評価対象となるバスに乗る変数 */

/* メイン関数 */
void main(void)
{
 int bus0;
 .
 .
 .
 a=0;                           /* バスに乗る変数に書き込みが発生した場合 */
 bus0=BUS0();                   /* 直後に評価関数を埋め込む */
 .
 .
 .
 b=1;              /* 変数bは、バスに乗らないので、評価関数は埋め込まない */
 .
 .
 .
 c=10;
 bus0=BUS0();
 .
 .
 .
 a=3;
 bus0=BUS();
 b=4;              /* 変数bは、バスに乗らないので、評価関数は埋め込まない */

 printf("BUS Traffic=%d[回]¥n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
 static int i=0;
 i++;                           /* スタティック変数iをインクリメント */
 return i;
}
```

【図４】


START
評価関数作成
C1
アルゴリズム設計で用いたソースを、構文解析を行いながら１行ずつ読み込む
C2
C3
評価対象となるパスに乗る変数？
NO
YES
C4
直前もしくは直後に評価関数を埋め込む
C5
NO
読み込んだソースの最後の行？
YES
コンパイル
C6
実行
C7
性能評価
C8
END

【図５】

```
#include <stdio.h>

int BUS0(void);

/* メイン関数 */
void main(void)
{
 int a;                          /* 評価対象となるバスに乗る変数 */
 int b;                          /* 評価対象となるバスに乗る変数 */
 int c;                          /* 評価対象となるバスに乗る変数 */

 int bus0;
 .
 .
 .
 a=0;                            /* バスに乗る変数に書き込みが発生した場合 */
 bus0=BUS0();                    /* 直後に評価関数を埋め込む */
 .
 .
 .
 b=1;
 bus0=BUS0();
 .
 .
 .
 c=10;
 bus0=BUS0();
 .
 .
 .
 a=3;
 bus0=BUS();
 b=4;
 bus0=BUS();

 printf("BUS Traffic=%d[回]¥n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
 static int i=0;                 /* スタティック変数iをインクリメント */
 i++;
 return i;
}
```

【図６】

```
#include <stdio.h>

int BUS0(void);

int a;                          /* 評価対象となるバスに乗る変数 */
int b;                          /* 評価対象となるバスに乗る変数 */
int c;                          /* 評価対象となるバスに乗る変数 */

/* メイン関数 */
void main(void)
{
 int bus0;                      /* バスに乗る変数に書き込みが発生した場合 */
 .                              /* 直前に評価関数を埋め込む */
 .
 .
 bus0=BUS0();
 a=0;
 .
 .
 .
 bus0=BUS0();
 b=1;
 .
 .
 .
 bus0=BUS0();
 c=10;
 .
 .
 .
 bus0=BUS ();
 a=3;
 bus0=BUS();
 b=4;

 printf("BUS Traffic=%d[回]¥n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
 static int i=0;
 i++;                           /* スタティック変数iをインクリメント */
 return i;
}
```

【図７】

```
#include <stdio.h>

int BUS1(int, int);

int a;                          /* 評価対象となるバスに乗る変数(32ビット) */
int b;                          /* 評価対象となるバスに乗る変数(16ビット) */

/* メイン関数 */
void main(void)
{
 int bus1;
 .
 .
 .
 a=7;                           /* バスに乗る変数に書き込みが発生した場合 */
 bus1=BUS1(32,8);               /* 直後に評価関数を埋め込む */
 .
 .
 .
 bus1=BUS1(32,8);
 a=6;
 bus1=BUS1(16,8);
 b=10;
 printf("BUS Traffic=%d[回]¥n", bus1);
}

/* 評価関数 */
int BUS1(int bit, int bus)      /* バスのデータ転送分インクリメントする関数 */
{
 static int i=0;
 i+=bit / bus;
 return i;
}
```

【図８】

```
#include <stdio.h>

int BUS2(int);

int a[10];                    /* 評価対象となるバスに乗る変数(配列) */
int b[20];                    /* 評価対象となるバスに乗る変数(配列) */

/* メイン関数 */
void main(void)
{
 int bus2;
 int *adr;
 .
 .
 .
 adr=&a[0];                   /* バスに乗る変数のアドレス転送が発生した場合 */
 bus2=BUS2(10);               /* 直後に評価関数を埋め込む */
 .
 .
 .
 adr=&a[0];
 bus2=BUS2(10);
 adr=&b[0];
 bus2=BUS2(20);
 printf("BUS Traffic=%d[回]¥n", bus2);
}

/* 評価関数 */
int BUS2(int ele)             /* 要素数分インクリメントする関数 */
{
 static int i=0;
 i+=ele;
 return i;
}
```

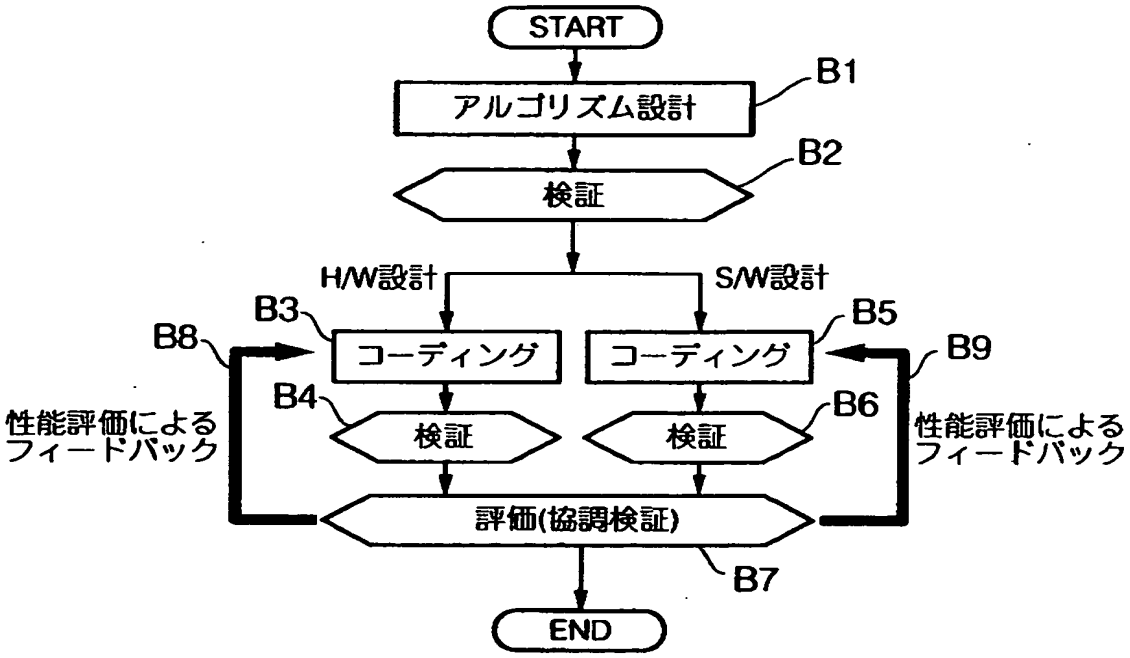
【図９】

```
#include <stdio.h>

void BUS3(void);

int a;                          /* 評価対象となるバスに乗る変数 */
int b;                          /* 評価対象となるバスに乗る変数 */

int bus=0;                      /* 評価関数内でインクリメントされる変数 */

/* メイン関数 */
void main(void)
{
 .
 .
 .
 a=0;                           /* バスに乗る変数に書き込みが発生した場合 */
 BUS3();                        /* 直後に評価関数を埋め込む */
 .
 .
 .
 a=1;
 BUS3();
 b=10;
 BUS3();
 printf("BUS Traffic=%d[回]¥n", bus);
}

/* 評価関数 */
void BUS3(vois)
{
 bus++;                         /* グローバル変数busをインクリメント */
}
```

【図１０】

```
#include <stdio.h>

/* BUSクラス */
classBUS
{
 int i;

public:
 BUS():i(0){}

 int count(){return i++;}                    /* iをインクリメントする関数 */
 int count_bit(int bit, int bus)             /* バスのデータ転送分インクリメントする関数 */
 {
  i+=bit/bus;
  return i;
 }
 int count_hairetu(int, ele)                 /* 要素分インクリメントする関数 */
 {
 i+=ele;
 return i;
 }

 void print (){printf("BUS Traffic=%d[回]¥n",i);}
 };

 int a;                                      /* bus0に乗る変数 */


 /* メイン関数 */
 int main()
 {
 BUS bus0;                                   /* 評価対象となるバス(32ビット) */
 BUS bus1;                                   /* 評価対象となるバス(8ビット) */
 BUS bus2;                                   /* 評価対象となるバス(32ビット) */

 int b;                                      /* bus1に乗る変数(32ビット) */
 int c[10];                                  /* bus2に乗る変数(配列) */

 int *cp;

 int i;

 a=0;                                        /* 評価対象となるバスに乗る変数にデータ書き込みが */
 bus0.count();                               /* 行われた場合、直後に評価関数を埋め込む */

 b=1;
 bus1.count_bit(32, 8)

 cp=&c[0];
 bus2.count_hairetu(10);

 for(i=0; i<10; i++){
  bus0.count();                              /* 評価対象となるバスに乗る変数にデータ書き込みが */
  a=i;                                       /* 行われた場合、直前に評価関数を埋め込む */

  bus1.count_bit(32, 8);
  b=i+1;

  bus2.count();
  c[i]=i*2;
 }

 bus0.print();
 bus1.print();
 bus2.print();

 return0;
 }
```

【図１１】


START
アルゴリズム設計
B1
検証
B2
H/W設計
S/W設計
B3
B8
コーディング
コーディング
B5
B9
B4
検証
検証
B6
性能評価による
フィードバック
性能評価による
フィードバック
評価(協調検証)
B7
END

【書類名】　　要約書

【要約】

【課題】　コーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排し、設計ターンアラウンドタイムの大幅な削減を行う。

【解決手段】　アルゴリズム設計で作成したＣ言語やＣ$^{++}$等汎用高級言語で記述されたソースをハードウェア、ソフトウェアに切り分けた後（ステップＡ３）、バスのトラフィックをカウントする評価関数を作成して（ステップＡ４）、バスにデータ(変数)が乗る毎に、その評価関数を実行するようにソースを修正し（ステップＡ６）、その後、性能評価を行い（ステップＡ７）、最終的に算出された処理レートに対するバストラフィック（ステップＡ８、Ａ９）により、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行う（ステップＡ１６）。

【選択図】　　図１

認定・付加情報

特許出願の番号	特願２０００－１６６６３０
受付番号	５００００６８９８０３
書類名	特許願
担当官	塩崎　博子　　１６０６
作成日	平成１２年　６月　９日

＜認定情報・付加情報＞

【特許出願人】
- 【識別番号】 000232036
- 【住所又は居所】 神奈川県川崎市中原区小杉町１丁目４０３番５３
- 【氏名又は名称】 日本電気アイシーマイコンシステム株式会社

【代理人】 申請人
- 【識別番号】 100108578
- 【住所又は居所】 東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所
- 【氏名又は名称】 高橋　詔男

【代理人】
- 【識別番号】 100064908
- 【住所又は居所】 東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所
- 【氏名又は名称】 志賀　正武

【選任した代理人】
- 【識別番号】 100101465
- 【住所又は居所】 東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所
- 【氏名又は名称】 青山　正和

【選任した代理人】
- 【識別番号】 100108453
- 【住所又は居所】 東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所
- 【氏名又は名称】 村山　靖彦

次頁無

出願人履歴情報

識別番号　〔０００２３２０３６〕

１．変更年月日　１９９０年　８月１３日
［変更理由］　新規登録
住　所　神奈川県川崎市中原区小杉町１丁目４０３番５３
氏　名　日本電気アイシーマイコンシステム株式会社